智慧化企業整合

project2

真的假不了—基於深度學習辨識面容真偽

Group 9

110034541 李顥廷

110034550 黃喆志

110034566 黃采琳

指導教授：邱銘傳 教授

目錄

一、背景介紹.....1
1-1. 情境描述.....1
1-2. 5W1H.....1
1-3. 計畫流程.....1
二、資料蒐集與整理.....2
2-1.資料來源.....2
2-2.資料前處理.....2
三、模型建構.....7
3-1. 模型介紹.....7
3-2. 模型建立與訓練.....9
四、參數優化.....11
4-1. 模型訓練.....11
4-2. 實驗設計方法介紹.....12
4-3. 參數優化結果.....12
4-4. Densenet 結果:.....14
五、結論.....15
5-1. 研究結果.....15
5-2. 研究限制.....16
5-3. 未來研究方向.....17

一、背景介紹

1-1. 情境描述

近幾年臉部特效帶給大眾許多新奇有趣的體驗，換臉 APP 讓大家可以用自拍照秒變大咖巨星、隨意轉換性別的長相、模擬老年模樣等特殊效果，Deepfake 一詞也隨之被提出。

Deepfake，結合了 Deep learning 與 fake 兩個詞彙，也就是人工智慧的人體圖像合成技術，只要有欲仿造對象的人物影音或影像素材，就能產出肉眼難以辨識的內容，達到以假亂真的境界。透過 Deepfake，我們可以「喚醒故人」，一間語音服務新創公司 Aflorithmic 曾釋出一段機器人的影片，影片中已故的物理學家愛因斯坦（Albert Einstein）彷彿像活過來一般，不只聲音重現，更使臉部表情完美呈現；另外 Deepfake 技術也能讓聾啞人獲得聲音，美國波士頓兒童醫院為一群不能說話的患者進行聲音再製，讓聾啞人依舊能用原聲與他人溝通。

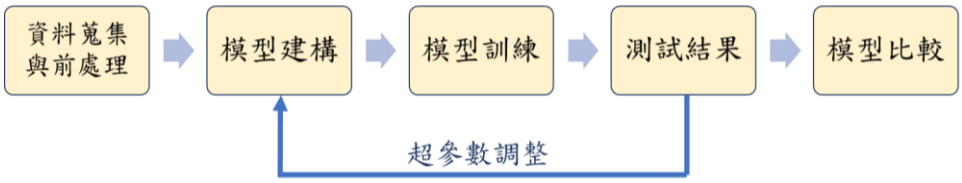
然而我們從近期的新聞可以得知，隨著此項技術門檻越來越低，將會被有心人士利用，進行惡意惡作劇或是製作假新聞來顛倒是非、混淆視聽；更甚是販賣不法影片賺取暴利，在這真真假假、資訊爆炸的時代，本小組利用 CNN 來進行真假臉辨識，期許未來能以此為架構建立一真假辨識系統。

1-2. 5W1H

Who	對圖片真偽有疑慮的群眾、偵辦相關案件的人員(警察)
What	辨認圖片人像的真偽
When	當看到圖片時不確定圖中人像的真偽
Where	每個地方
Why	臉部合成技術越來越發達，可防範不法用途
How	深度學習、機器學習、資料分析

1-3. 計畫流程

本研究的流程如下圖。首先蒐集真假臉的圖片，並針對所蒐集的圖片進行前處理，接著建構模型，並對其訓練，接著測試模型的訓練結果，並針對本研究建立的模型進行模型比較。



二、資料蒐集與整理

2-1. 資料來源

資料源自於 kaggle 網站的 140k Real and Fake Face 資料集，顧名思義此資料集包含 14 萬張圖片，其中有 7 萬張圖片為真實人臉，是從 Flickr 的數據集收集而來的；另外 7 萬張假臉是由 Bojan 利用 StyleGAN 生成而來的。此資料集有一大優勢為他已事先將所有數據拆分為訓練集、驗證集和測試集，幫助本研究能更聚焦於模型的訓練，並找出更好的模型。

2-2. 資料前處理

2-2-1. Colab 導入資料集

將資料集上傳雲端於 colab 進行壓縮

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from zipfile import ZipFile
file_unzip = 'drive/MyDrive/real-vs-fake.zip'
with ZipFile(file_unzip, 'r') as zip:
    zip.extractall()
    print("done")
```

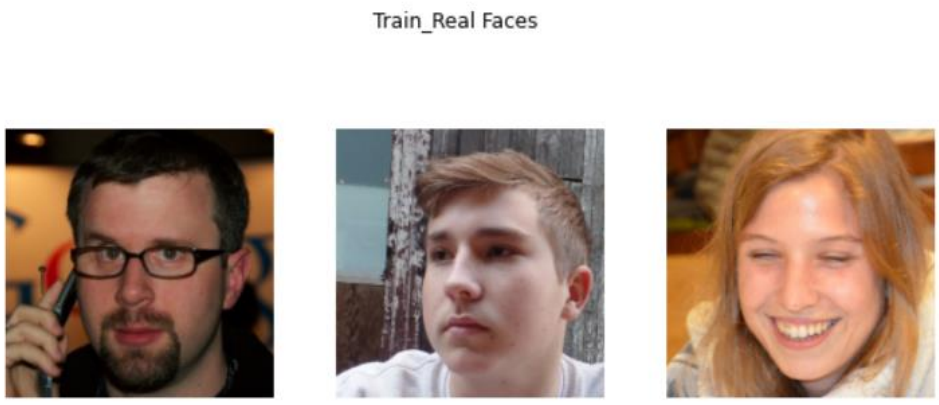
```
done
```

2-2-2. 載入相關套件

```
import numpy as np
import pandas as pd
import os
import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.layers import Conv2D, Dense, BatchNormalization, MaxPooling2D, GlobalAveragePooling2D, Dropout, Flatten

from tensorflow.keras.optimizers import RMSprop,SGD,Adam,Adagrad
from keras import regularizers
from tensorflow.keras.applications import MobileNetV2
import matplotlib.pyplot as plt
from tensorflow.keras.utils import plot_model
```

2-2-3. 建立目錄及視覺化訓練及測試集

```
path = '../content/real-vs-fake'
train = '../content/real-vs-fake/train'
test = '/content/real-vs-fake/test'
val = '../content/real-vs-fake/valid'
train_fake = '/content/real-vs-fake/train/fake'
train_real = '/content/real-vs-fake/train/real'
test_fake = '/content/real-vs-fake/test/fake'
test_real = '/content/real-vs-fake/test/real'

def plot_image(path, title):
        plt.figure(figsize=(10,10))
        for i in range(9):
                img = load_img(path +'/' +os.listdir(path)[i])
                plt.subplot(3,3,i+1)
                plt.imshow(img)
                if title=='Fake Faces':
                        plt.title(os.listdir(path)[i][:4])
                plt.suptitle(title)
                plt.axis('off')
        return plt

plot_image(train_real, 'Train_Real Faces').show()
plot_image(train_fake, 'Train_Fake Faces').show()

plot_image(test_real, 'Test_Real Faces').show()
plot_image(test_fake, 'Test_Fake Faces').show()
```



Train_Fake Faces



Test_Real Faces



Test_Fake Faces



2-2-4. 更改圖檔名稱

原先圖檔命名皆為數字編號，為方便辨識，圖片檔名加入 real 或 fake

```
##輸入新的檔名
newname = "real"
newname = newname.strip()
if newname != '':
        ##獲取當前資料夾的路徑
        train_real = os.getcwd()
        ##要修改的檔案的格式
        pic_ext = ['.jpg','.png']
        i = 0
        for file in os.listdir(path):
                if os.path.isfile(file) == True:
                        name,ext = os.path.splitext(file)
                        print("ext")
                        if ext in pic_ext:
                                i = i+1
                                newname1 = newname + '_' + str(i) + ext
                                os.rename(file,newname1)

##輸入新的檔名
newname = "fake"
newname = newname.strip()
if newname != '':
        ##獲取當前資料夾的路徑
        train_fake = os.getcwd()
        ##要修改的檔案的格式
        pic_ext = ['.jpg','.png']
        i = 0
        for file in os.listdir(path):
                if os.path.isfile(file) == True:
                        name,ext = os.path.splitext(file)
                        print("ext")
                        if ext in pic_ext:
                                i = i+1
                                newname1 = newname + '_' + str(i) + ext
                                os.rename(file,newname1)
```

2-2-5. 將圖像歸一化

使用 ImageDataGenerator 來讓 train、validation、test 的像素值從[0-255]壓縮至[0-1]，其中 flow_from_directory 來指定對應的資料夾，class_mode 設為 binary、圖片大小 target_size 設為(224,224)，shuffle 設定是否打亂數據，batch_size 設為 64。

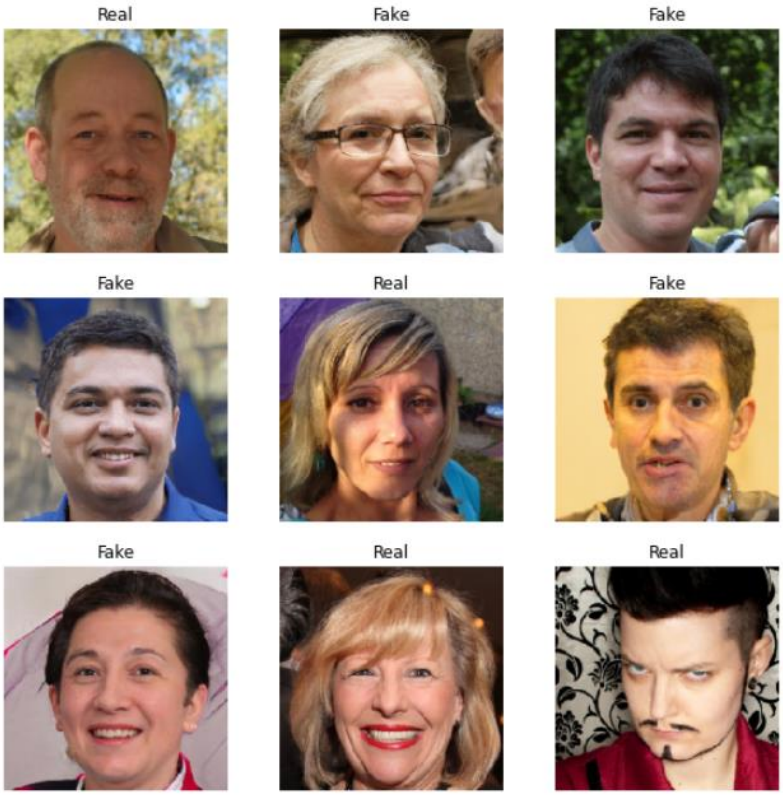
程式結果呈現 train 共有 100000 筆、validation 共有 20000 筆、test 共有 20000 筆，分為兩個 classes，fake 代表 0、real 代表 1。

```
datagen = ImageDataGenerator(rescale=1./255)

train_set = datagen.flow_from_directory(train,
                                        class_mode='binary',
                                        shuffle=True,
                                        target_size=(224,224),
                                        batch_size=64
                                       )
validation_set = datagen.flow_from_directory(val,
                                             class_mode='binary',
                                             shuffle=True,
                                             target_size=(224,224),
                                             batch_size=64
                                            )
test_set = datagen.flow_from_directory(test,
                                       class_mode='binary',
                                       shuffle=True,
                                       target_size=(224,224),
                                       batch_size=64
                                      )

train_set.class_indices
```

```
Found 100000 images belonging to 2 classes.
Found 20000 images belonging to 2 classes.
Found 20000 images belonging to 2 classes.
{'fake': 0, 'real': 1}
```

2-2-6. 視覺化呈現真假圖片標籤

```
plt.figure(figsize=(10, 10))
for i in range(9):
        img, label = train_set.next()
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(img[0])
        if(label[0] == 0.0):
                plt.title("Fake")
        else:
                plt.title("Real")
        plt.axis("off")
```



三、模型建構

3-1. 模型介紹

3-1-1. CNN

卷積神經網絡(Convolutional Neural Network)簡稱 CNN，其在影像識別上具有非常良好的成效，許多影像辨識模型也是以 CNN 為架構進行延伸，另外 CNN 也是少數參考人類大腦視覺組織來建立的深度學習模型，其架構會包含：

1. 卷積層 Convolution Layer

將原始圖片與特定的 Feature Detector(filter)做卷積運算

2. 池化層 Pooling Layer

主要採用 Max pooling， Max pooling 的好處是當整張圖片平移幾個 Pixel 對判斷上完全不會造成影響，且具有很好的抗雜訊功能。

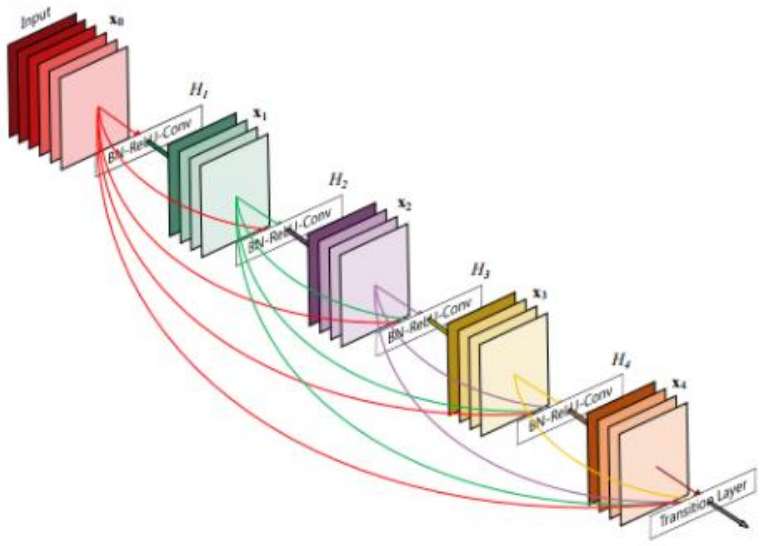
3. 全連接層 Fully Connected Layer

將先前的結果拉直，就可以接到基本的神經網路，之後即可進行分類。

3-1-2. Densenet

在本小組研究的過程中，發現由 CNN 所延伸發展出的稠密卷積網路(Densenet)，此網路不是用很深或很寬的網路來獲得呈現圖像辨識的能力，他是以前饋方式，將每層連結到每個其它層，不像傳統具有 L 層的卷積網絡有 L 個連接，而是每個層與其後一個層之間，又有 L(L+1)/2 個直接連接，並透過特徵的重複使用來得到網絡的隱含信息，獲得更容易訓練、參數效率更高的稠密模型。

Dense block 的結構圖：



每一層的輸入來自前面所有層的輸出。就是說 x_0 是 input，H_1 的輸入是 x_0（input），H_2 的輸入是 x_0 和 x_1 （x_1 是 H1 的輸出）。

Densenet 模型架構：

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

總結：

Densenet 的優點包含緩解梯度消失問題、加強特徵傳播、大幅減少參數數量；缺點為訓練的過程很佔記憶體，在本研究利用此方法訓練的過程中也有明顯的感受到訓練時間冗長的狀況。

3-2. 模型建立與訓練

3-2-1. CNN

建立一卷積神經網路模型：

(1) 建立卷積層(filter = 32)

(2) 建立池化層

(3) 建立卷積層(filter = 64)

(4) 建立池化層

(5) 建立卷積層(filter = 128)

(6) 建立池化層

(7) 進行 flatten 拉直成 1-D 陣列

(8) 輸出 fake 或 real 兩類

Activation function 採用 relu。

```
model = tf.keras.models.Sequential(
    [
      tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(224, 224, 3)),
      tf.keras.layers.MaxPooling2D(2,2),

      tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
      tf.keras.layers.MaxPooling2D(2,2),

      tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'),
      tf.keras.layers.MaxPooling2D(2,2),

      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(2)
    ])
model.summary()
model.compile(optimizer= Adam(lr=0.001), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics = ['accuracy'])
```

3-2-2. Densenet

建立一稠密卷積神經網路(Densenet)模型:

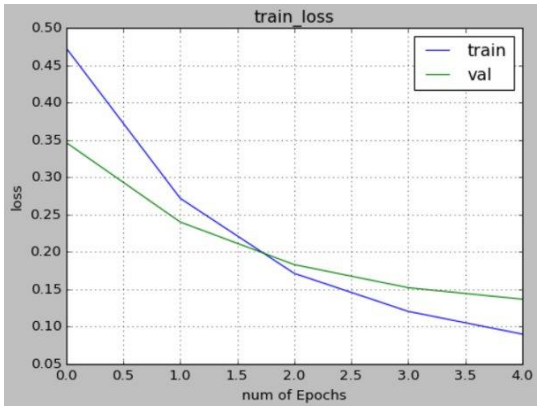
```
#Creating  Model
from  tensorflow.keras.applications  import  DenseNet121
def  get_model():
        densenet  =  DenseNet121(weights='imagenet',
                                 include_top=False,
                                input_shape=(224,224,3)
                                )
        model  =  tf.keras.models.Sequential([densenet,
                                              GlobalAveragePooling2D(),
                                              Dense(512,  activation='relu'),
                                              BatchNormalization(),
                                              Dropout(0.3),
                                              Dense(1,  activation='sigmoid')
                                              ])
        model.compile(optimizer=Adam(lr=0.001),
                                loss='binary_crossentropy',
                                metrics=['accuracy']
                                )

        return  model

spoofnet  =  get_model()
spoofnet.summary()
```

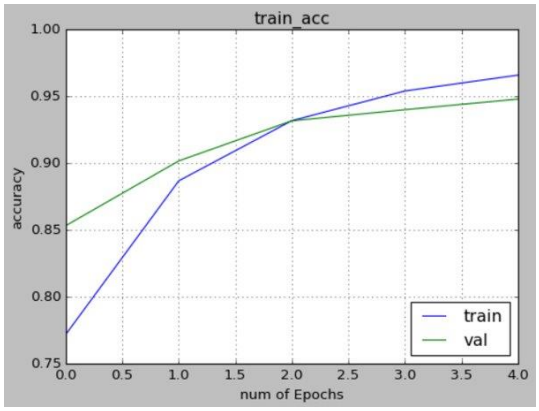
此模型採用 Densenet121 架構，利用 relu 及 sigmoid 作為激活函數，來幫助本研究分辨真假臉。

四、參數優化

4-1. 模型訓練

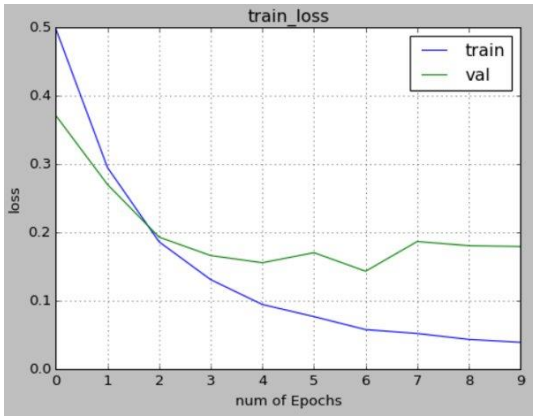
以 3-2-1 的 CNN 模型，嘗試先跑 epoch = 5 來看訓練結果：

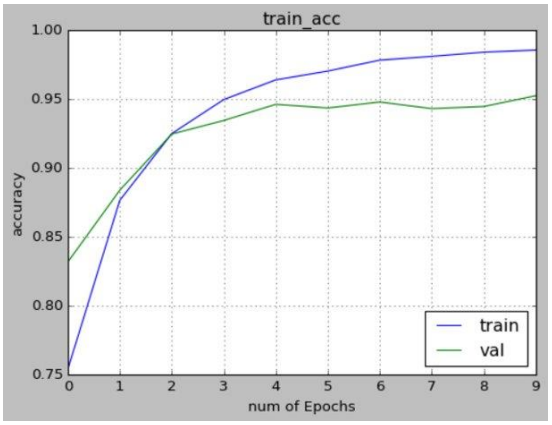




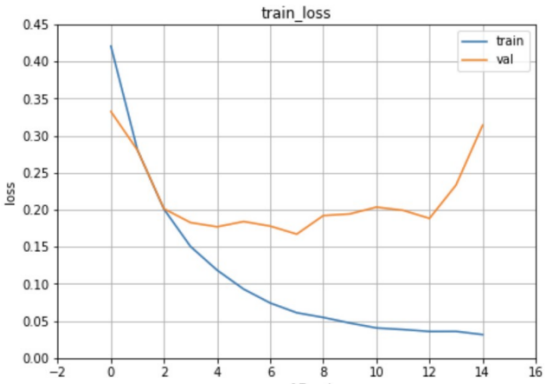
接著嘗試增加 epoch 進行訓練，並在每一個 epoch 結束時計算驗證集（validation）的準確率，當準確率不再提高就停止訓練，以確定最佳的訓練 epoch。

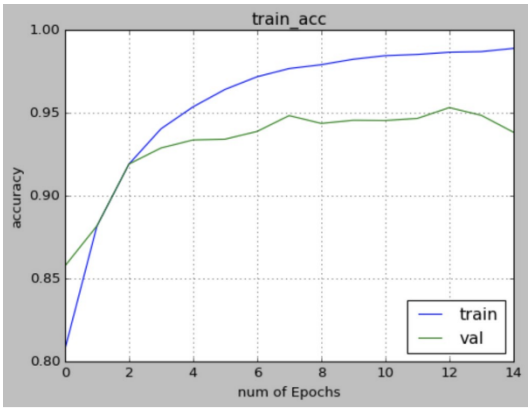
當 epoch = 10：





當 epoch = 15：





由上圖結果可以發現，在 epoch=12 之後，validation 的 loss 越來越高，具有明顯的 overfitting，因此本研究後續進行模型提升、超參數優化，都將 epoch 設為 12 來訓練。

4-2. 實驗設計方法介紹

實驗設計是統計學的一種方法，透過合理地挑選試驗條件，並通過對試驗資料的分析，從而建立水準與因數之間的函數關係，找出總體最優的改進方案。實驗設計最大的目的就是能減少試驗次數，本研究欲利用此優勢來對模型的超參數調整，藉此能在有科學依據下，更有效率地找出最適合的模型參數組合。

下圖為本研究所考慮的因子及水準:

Factor \ Level	Level 1	Level 2	Level 3
Learning rate	0.005	0.001	0.0001
Optimizer	Adam	SGD	Adagrad
Activate function	Relu	Tanh	sigmoid

4-3. 參數優化結果

	Learning rate	Optimizer	Activate function	train acc	test acc
0	0.001	Adam	Relu	0.9383	0.939
1	0.005	Adam	Relu	0.5004	0.5
2	0.005	SGD	sigmoid	0.5722	0.5928
3	0.005	Adagrad	Tanh	0.9663	0.9319

4	0.001	Adam	sigmoid	0.8878	0.9417
5	0.001	SGD	Tanh	0.7998	0.7962
6	0.001	Adagrad	Relu	0.8175	0.8141
7	0.0001	Adam	Tanh	0.9934	0.9620
8	0.0001	SGD	Relu	0.6325	0.6434
9	0.0001	Adagrad	sigmoid	0.4952	0.5014

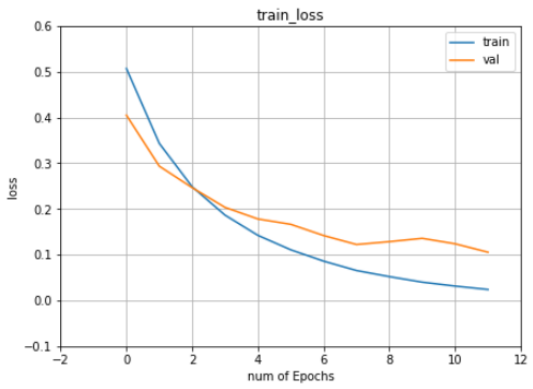
以下為實驗七的訓練過程：

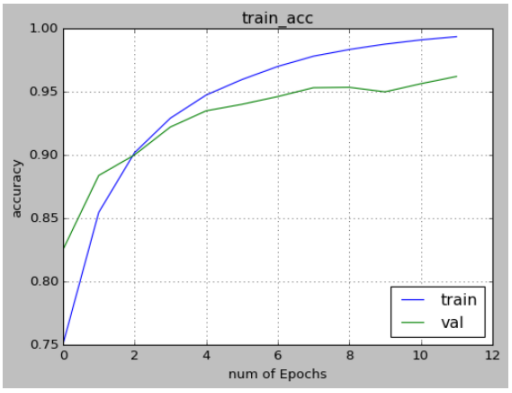
```
history = model.fit(train_set, validation_data = validation_set, epochs = 12, validation_steps = 100)
```

```
Epoch 1/12
1563/1563 [==============================] - 418s 260ms/step - loss: 0.5072 - accuracy: 0.7501 - val_loss: 0.4051 - val_accuracy: 0.8250
Epoch 2/12
1563/1563 [==============================] - 344s 220ms/step - loss: 0.3434 - accuracy: 0.8545 - val_loss: 0.2936 - val_accuracy: 0.8838
Epoch 3/12
1563/1563 [==============================] - 340s 217ms/step - loss: 0.2485 - accuracy: 0.9017 - val_loss: 0.2467 - val_accuracy: 0.9000
Epoch 4/12
1563/1563 [==============================] - 337s 216ms/step - loss: 0.1866 - accuracy: 0.9290 - val_loss: 0.2035 - val_accuracy: 0.9220
Epoch 5/12
1563/1563 [==============================] - 345s 220ms/step - loss: 0.1423 - accuracy: 0.9473 - val_loss: 0.1782 - val_accuracy: 0.9348
Epoch 6/12
1563/1563 [==============================] - 336s 215ms/step - loss: 0.1107 - accuracy: 0.9596 - val_loss: 0.1665 - val_accuracy: 0.9400
Epoch 7/12
1563/1563 [==============================] - 339s 216ms/step - loss: 0.0859 - accuracy: 0.9699 - val_loss: 0.1419 - val_accuracy: 0.9461
Epoch 8/12
1563/1563 [==============================] - 340s 218ms/step - loss: 0.0654 - accuracy: 0.9780 - val_loss: 0.1223 - val_accuracy: 0.9531
Epoch 9/12
1563/1563 [==============================] - 338s 216ms/step - loss: 0.0521 - accuracy: 0.9833 - val_loss: 0.1286 - val_accuracy: 0.9534
Epoch 10/12
1563/1563 [==============================] - 340s 217ms/step - loss: 0.0400 - accuracy: 0.9876 - val_loss: 0.1358 - val_accuracy: 0.9498
Epoch 11/12
1563/1563 [==============================] - 340s 218ms/step - loss: 0.0314 - accuracy: 0.9909 - val_loss: 0.1239 - val_accuracy: 0.9563
Epoch 12/12
1563/1563 [==============================] - 346s 221ms/step - loss: 0.0239 - accuracy: 0.9934 - val_loss: 0.1055 - val_accuracy: 0.9620
```

```
accu = model.evaluate(test_set)
```

```
313/313 [==============================] - 68s 218ms/step - loss: 0.1115 - accuracy: 0.9605
```



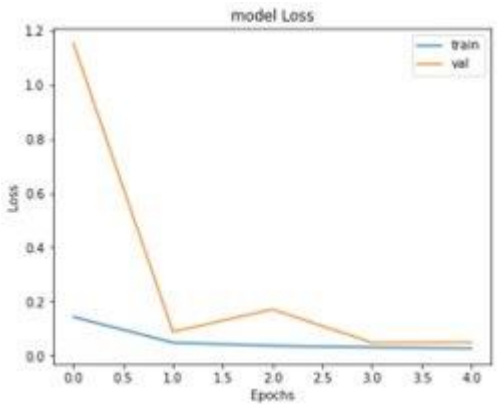


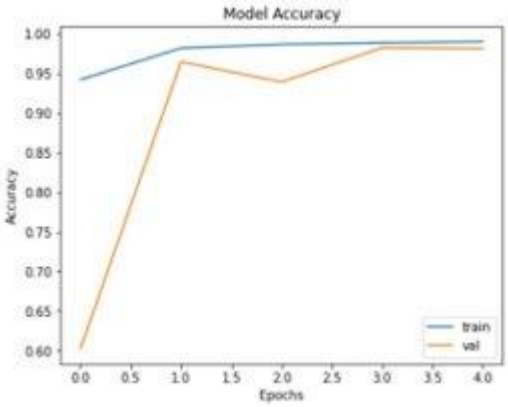
4-4. Densenet 結果:

下圖為利用 Densenet 模型訓練的過程以及結果，從圖中可以得知一次 epoch 所需的時間大約為 35 分鐘，訓練的準確度從第一次的 0.9422 到第五次達到 0.99，而驗證的準確度也有隨之上升，來到 0.9817，並未有顯著的 overfitting 的情形。

```
history = spoofnet.fit(train_set, validation_data = validation_set, epochs = 5, validation_steps = 100)
```

```
Epoch 1/5
1563/1563 [==============================] - 2090s 1s/step - loss: 0.1429 - accuracy: 0.9422 - val_loss: 1.1515 - val_accuracy: 0.6030
Epoch 2/5
1563/1563 [==============================] - 2043s 1s/step - loss: 0.0487 - accuracy: 0.9821 - val_loss: 0.0885 - val_accuracy: 0.9647
Epoch 3/5
1563/1563 [==============================] - 2042s 1s/step - loss: 0.0368 - accuracy: 0.9867 - val_loss: 0.1711 - val_accuracy: 0.9392
Epoch 4/5
1563/1563 [==============================] - 2052s 1s/step - loss: 0.0310 - accuracy: 0.9886 - val_loss: 0.0484 - val_accuracy: 0.9825
Epoch 5/5
1563/1563 [==============================] - 2053s 1s/step - loss: 0.0268 - accuracy: 0.9900 - val_loss: 0.0494 - val_accuracy: 0.9817
```





下圖為 Densenet 的測試結果，可以得到 0.983 的準確率，這樣的結果相較前述利用 CNN 的模式有更佳的準確率，對比之下，利用此方法進行訓練可以達到更好的效果，這也是未來相關研究可以努力的方向。

```
#Accuracy On test set
_, accu = spoofnet.evaluate(test_set)
print('Final Test Acccuracy = {:.3f}'.format(accu*100))
```

```
313/313 [==============================] - 113s 359ms/step - loss: 0.0477 - accuracy: 0.9830
Final Test Acccuracy = 98.300
```

五、結論

5-1. 研究結果

本研究探討了 CNN 模型和 Densenet 模型在訓練上的差異，並透過實驗設計的方法確認了調整超參數對訓練集正確率、測試集正確率的影響，結果表明將學習率設為 0.0001，Optimizer 設為 Adam， Activate function 採用 Tanh 能得到最好的結果。下表為本研究所使用的兩種模型比較表，包含兩種模型的特點、差異、訓練過程的時間以及訓練結果準確率的情況。

	CNN	Densenet
全名	卷積神經網路	稠密卷積神經網路
特點	透過卷積和池化層，增進細節辨識能力	加強特徵傳播，特徵重用、減緩梯度消失問題
連接數(L 層)	L 個連接	L（L+1）/ 2 個連接
訓練時間	較短	較長
訓練結果	準確率較低(0.9620)	準確率較高(0.983)

另外也對模型的泛化能力進行測試，輸入非訓練及驗證集的圖片，讓模型判斷輸入模型的照片有多少機率是真、多少機率是假。下圖為一張 fake 的照片，而模型判斷有 99.44%為假。

```
test_image = image.load_img('/content/real-vs-fake/test/fake/0HOBK2WCQQ.jpg', target_size=(224, 224, 3))
plt.imshow(test_image)

test_image_arr = image.img_to_array(test_image)
test_image_arr = np.expand_dims(test_image, axis=0)

logits = model.predict(test_image_arr)
predictions = tf.nn.softmax(logits).numpy()
print(predictions)

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(predictions)], 100 * np.max(predictions))
)

#print(100 * np.max(predictions))

if np.argmax(predictions) == 0:
    title = "fake prob: " + str(100 * np.max(predictions))
else:
    title = "real prob: " + str(100 * np.max(predictions))

plt.title(title)
plt.imshow(test_image)
```

```
[[0.99444735 0.0055527 ]]
This image most likely belongs to fake with a 99.44 percent confidence.
<matplotlib.image.AxesImage at 0x7f5fec518c10>
```

5-2. 研究限制

本研究的面容真偽辨識僅限於照片，顯然在現實世界僅辨識照片真偽有泛用性不足的問題，因為影片比起圖片往往更有混淆大眾視聽的影響力。

5-3. 未來研究方向

未來能將本研究與網站、chatbot 做結合，大眾若對收到的圖像有疑慮即可將圖像上傳到所搭建的網站或 chatbot，即時確認圖像的真偽。

另外由 Densenet 所訓練的模型結果發現，其測試精確度高達 98.3%，礙於時間因素無法進一步進行模型參數優化及研究，也許未來能以 Densenet 來進行訓練得到更好的效果。